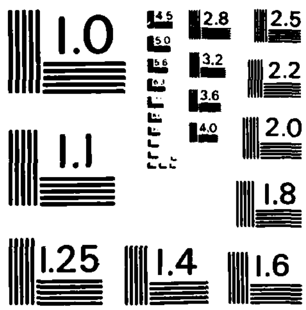
AD-A130 905 DYNAMIC PROGRAMMING ALGORITHMS AND ANALYSES FOR NONSERIAL NETWORKS PART II(U) ATLANTA UNIV GA DEPT OF MATHEMATICAL SCIENCES N A WARSI JAN 83 1/1

UNCLASSIFIED ARO-17672.2-MA-H DAAG29-80-G-0010 F/G 12/1 NL

END
DATE
FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(12)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 17672.2-MA-H
2. GOVT ACCESSION NO.: AD-A130905
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Dynamic Programming Algorithms and Analyses for Nonserial Networks: Part II
5. TYPE OF REPORT & PERIOD COVERED: Final: 25 Sep 80 - 24 Sep 83
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Nazir A. Warsi
8. CONTRACT OR GRANT NUMBER(s): DAAG29 80 G 0010
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Atlanta University, Atlanta, GA 30314
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: U. S. Army Research Office, Post Office Box 12211, Research Triangle Park, NC 27709
12. REPORT DATE: Jan 83
13. NUMBER OF PAGES: 42
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

AUG 1 1983

A

18. SUPPLEMENTARY NOTES

The view, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

computer programming
nonserial networks
operations research
dynamic programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The design of algorithm plays an important role in operations research in general and dynamic programming in particular. In most dynamic programming algorithms, formalism of computing, data structure and complexity analysis does not appear. One of the objectives of this research is to provide such a formalism. Converging branch, diverging branch, feed-forward loop and feedback loop systems are considered. In each case, first, a high level algorithm followed by the detailed computer algorithm is described. Formulas for storage and computational complexities for each computer algorithm are derived. Finally algorithms are implemented on VAX-11/780 computers using UCSD PASCAL.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A130905

DTIC FILE COPY

DYNAMIC PROGRAMMING ALGORITHMS AND ANALYSES

FOR

NONSERIAL NETWORKS: PART II

By

Nazir A. Warsi

Completion Report

A.U. Project No. 80-028(2)

ARO No. DAAG 29-80-G-0010

Initiated: October 1980

Completed: January 1983

The work upon which this report is based was supported by the U.S. Army Research Office and the Atlanta University.

Department of Mathematical Sciences

Atlanta University

Atlanta, Georgia 30314

A

I. Introduction

The design of algorithm plays an important role in operations research in general and dynamic programming in particular. In most dynamic programming algorithms, formalism of computing, data structure and complexity analysis does not appear. One of the objectives of this research is to provide such a formalism.

The storage and computational requirements of dynamic programming algorithms have limited their practical uses. The storage and computational demands can become excessive whenever the state vectors are of dimension 3 or more, or if the number of states grow exponentially. Several schemes have been proposed to reduce the dimensionality in dynamic programming. However, practical use of any algorithm dictates an efficient implementation of techniques on computers. But how efficiently do these algorithms run on computers necessitates their study from the point of view of complexity analysis, for an algorithm's performance profile is measured in terms of computing time and space that are consumed. The asymptotic complexity of an algorithm determines the size of a problem that can be solved by the algorithm. If input has size k, and for some constant c the algorithm solves the problem in ck^p time or takes ck^p storage spaces, we say that its computational/storage complexity is $c(k^p)$.

In subsequent sections, we consider converging branch, diverging branch, feed-forward loop and feedback loop systems. In each case, first, a high level algorithm followed by the detailed computer algorithm is described. Formulas for storage and computational complexities for each computer algorithm are derived. Finally algorithms are implemented on VAX-11/780 computers using UCSD PASCAL.

2. <u>Diverging Branch System</u>

$$\begin{array}{ccccccc} & d_{m1} & & & d_{11} & \\ x_{m1} & \boxed{m1} & x_{m-1,1} & x_{11} & \boxed{11} & x_{01} \\ & r_{m1} & & & r_{11} & \end{array}$$

$$\begin{array}{cccccccccc} & d_n & & & & d_s & & & d_1 & \\ x_n & \boxed{n} & x_{n-1} & \cdots\cdot & x_s & \boxed{s} & x_{s-1} & x_1 & \boxed{1} & x_0 \\ & r_n & & & & r_s & & & r_1 & \end{array}$$

Consider the diverging branch system with the following characteristics.

The state and decision variables are given by

$1 \leq x_{i1} \leq k_{i1}$, $1 \leq d_{i1} \leq p_{i1}$ where $1 \leq i \leq m$

$1 \leq x_i \leq k_i$, $1 \leq d_i \leq p_i$ where $1 \leq i \leq n$.

The transformations and returns are described by the following tables.

$x_{i-1} = t_i(x_i,d_i)$, $r_i = r_i(x_i,d_i)$ where $1 \leq i \leq n$

$x_{i-1,1} = t_{i1}(x_{i1},d_{i1})$, $r_{i1} = r_{i1}(x_{i1},d_{i1})$ where $1 \leq i \leq m$

$x_{m1} = t_{s1}(x_s,d_s)$

Let $k = \max(k_{11},\ldots k_{m1},k_1\ldots k_n)$ and $k_d = k + 1$.

2.1 <u>High-level Algorithm for Diverging Branch</u>

First process the branch including stages 11 through m1, and store the optimal return from stage m1. Process stages 1 through s-1 using serial DP-algorithm. Combine the optimal returns from stages s-1 and m1 to the return at stage s, and optimize the resultant return. Finally, process the stages s+1 through n using serial DP-algorithm and determine the optimal return at stage n.

A. For the branch including stages 11 through m1:

1. $f_{11}(x_{11}) = \max_{d_{11}} r_{11}(x_{11},d_{11});$

   Store d^*_{11}, the optimal decision.

2. $f_{i1}(x_{i1}) = \max_{d_{i1}}(r_{i1}(x_{i1},d_{i1}) + f_{i-1,1}\,(t_{i1}(x_{i1},d_{i1})));$

   Store d^*_{i1}, the optimal decision for $i = 2,\ldots,m$.

3. Store $f_{m1}(x_{m1})$ for later use.

B. For main branch including stages 1 through s-1:

1. $f_1(x_1) = \max_{d_1} r_1(x_1,d_1);$

   Store d^*_1, the optimal decision.

2. $f_1(x_1) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)));$

   Store the optimal decision d_i^* for $2 \leq i \leq s-1$.

C. For junction stage s:

$$f_s(x_s) = \max_{d_s}(r_s(x_s,d_s) + f_{s-1}(t_s(x_s,d_s)) + f_{m1}(t_{s1}(x_s,d_s)));$$

Store the optimal decision d_s^*.

D. For the main branch including stages s + 1 through n:

$$f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)));$$

Store the optimal decision d_i^* for $s + 1 \leq i \leq n$.

E. For the optimal return at stage n:

$$f_n(x_n^*) = \max_{x_n} f_n(x_n).$$

F. For the optimal decisions for stages n through 1:

For i = 1 to n:

Find d_i^*,

$x_{i-1}^* = t_i(x_i^*,d_i^*)$,

If $i = s + 1$ then $x_s = x^*_s$.

G. For optimal decisions for stages m_1 through 11:

$x_{m1}* = t_{s1}(x_s, d_s*)$,

For $i = m$ down to m_1:

Find $d_{i1}*$;

$x_{i-1,1} = t_{i1}(x_{i1}*, d_{i1}*)$;

2.2 Diverging Branch Computer Algorithm (DCBA)

In the above high-level algorithm, a brute-force method would be used to store optimal decisions at each stage for later retracing them after the optimal return $f_n(x_n*)$ has been found. This dictates an enormous amount of storage. Instead, a technique has been used, which marks the optimal decision value, say, d_i* by adding $k_d = 1 + k$ to the state entry $t_i(x_i*, d_i*)$, where $k = \max(k_{11}, \ldots k_{m1}, k_1, \ldots k_n)$. This eliminates the need for the optimal decision storage. Later when the optimal decision d_i* is to be retraced, it may be retrieved by searching x_i*-row of t_i for $t_0(x_i*, d_i*) \geq k_d$ over values of d_i. Any future reference to the table t_i will be made as (each entry) mod $2k_d$.

In a diverging branch system, tables F(0;1,1:K) are needed for optimal return processing at each stage. At each stage i the optimal return is processed by using the previous-stage optimal return from F(st,.), and stored in F(dt,.), where st = (i-1)mod 2, and dt = i mod 2. The optimal return f_{m1} is stored in the table FM(1:K). Note st and dt indicate indices of source and destination tables respectively.

We first explain the notations. The algorithms are described in a PASCAL-type construct. Comments are enclosed within (*...*). Enclosure of a simple or a complex statement within a loop is effected by indenting the statement. In the following, the computer algorithm for the

diverging branch system is described.

A. (* Branch including stages 11 through m1 *)

1. (* Initialize *)

For x_{11} = 1 to k_{11} do

$$F(1,x_{11}) = r_{11}(x_{11},d_{11}^*) = \max_{1 \le d_{11} \le p_{11}} r_{11}(x_{11},d_{11});$$

$$t_{11}(x_{11},d_{11}^*) = t_{11}(x_{11},d_{11}^*) + k_d;$$

2. (* Process stages 11 through m1 *)

```
For i = 2 to m do                  (* Find source table *)
     st = (i-1) mod 2;             (* Find destination table *)
     dt = i mod 2;
```

For x_{i1} = 1 to k_{i1} do

$$F(dt, x_{i1}) = r_{i1}(x_{i1}, d_{i1}^*) + F(st,t_i,(x_{i1},d_{i1}^*));$$

$$= \max_{1 \le d_{i1} \le p_{i1}} (r_{i1}(x_{i1},d_{i1})) + F(st,t_{i1}(x_{i1},d_{i1})));$$

$t_{i1}(x_{i1},d_{i1}^*) = t_{i1}(x_{i1},d_{i1}^*) + k_d$; (* mark optimal decision in t_{i1} *)

3. (* Move the optimal return $f_{m1}(x_{m1})$ to the table FM(.) and free F(0,.) and F(1,.) for further processing *)

For x_{m1} = 1 to k_{m1} do

$FM(x_{m1}) = F(dt,x_{m1});$

B. (* stages 1 through s-1 in the main branch *)

1. (* Initialize *)

```
For i = 1 to k do
     F(0,i) = 0;
```

2. For i = 1 to s-1 do

   st = (i-1) mod 2;

   dt = i mod 2;

      For x_i=1 to k_i do

$$F(dt, x_i) = r(x_i,d_i^*) + F(st,t_i(x_i,d_i^*)) = \max_{1 \le d_i \le p_i} (r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d$$

C. (* Junction s *)

   st = (s-1) mod 2;

   dt = s mod 2;

   For x_s = 1 to k_s do

$$F(dt,x_s) = r_s(x_s,d_s^*) + F(st,t_s(x_s,d_s^*)) + FM(t_{s1}(x_s,d_s^*)) = \max_{1 \le d_s \le p_s} (r_s(x_s,d_s) + F(st,t_s(x_s,d_s)) + FM(t_{s1}(x_s,d_s)));$$

$$t_s(x_s,d_s^*) = t_s(x_s,d_s^*) + k_d;$$

D. (* Stages s + 1 through n *)

   For i = s + 1 to n do

   st = (i-1) mod 2;

   dt = i mod 2;

   For x_i=1 to k_i do

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i^*)) = \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d;$$

E. (* Optimal return $f_n(x_n^*)$ *)

$$FN(x_n^*) = F(dt,x_n^*) = \max_{1 \le x_n \le k_n} F(dt,x_n);$$

F. (* Optimal decisions, stages n through 1 *)

For i=n down to 1 do

select x*-row of t_i and search for $t_i(x_i^*, d_i^*) \geq k_d$ for $1 \leq d_i \leq p_i$;

If i=s then $xs = x_s^*$ and $DS = d_s^*$

$x_{i-1}^* = t_i(x_i^*, d_i^*) \bmod k_d$;

G. (* Optimal decisions, stages m_1 through 11 *)

$x_{m1}^* = t_{s1}(XS, DS)$

For i=m down to 1 do

select x_{i1}^*-row of t_{i1} and search for $t_{i1}(x_{i1}^*, d_{i1}^*) \geq k_d$ for $1 \leq d_{i1} \leq p_i$

$x_{i-1,1}^* = t_{i1}(x_{i1}^*, d_{i1}^*) \bmod k_d$;

2.3 Storage Complexity of DBCA

We assume that each variable takes a unit storage space. We calculate the demand for the storage tables during computation. All input tables and variables created during the processing are ignored for this analysis.

Theorem: The storage complexity of DBCA is given by 3k or O(k)

Proof: The tables needed by DBCA are F(0:1,1;k) and FM(1:k) which consume 2k and k spaces respectively.

2.4 Computational Complexity of DBCA

For the purposes of our analysis, we consider all arithmetic operations, assignments, comparison and mod operations as basic computational operations. Other complex operations such as finding an optimum of a list or searching a list involves these basic operations. The computational complexity of an algorithm is given by the number of basic operations performed.

As an example consider a list $c_1, c_2, \ldots c_l$ of values where c_i is a composition of q of these basic operations. A simple routine to find the optimum value V will be:

$V = c_1$;

$J_o = 1$;

For j = n to 1 do

  If $V \leq C_j$, then $J_o = j$; $V = C_j$

The maximum number of the basic operations in this routine is $\sum_{j=1}^{l} (q_j+3)-1$.

Similarly, searching a value V in the list can take a maximum of l comparisons. Of course, more efficient searching algorithms are available. However, for the purposes of our analysis we choose this brute-force method to get an estimate on the worst cases.

In the following we derive the formula for computational complexity of DBCA.

<u>Theorem 2</u>: The computational complexity of DBCA is given by

$$\sum_{i=1} (4k_{i1}p_{i1} + k_{i1} + p_{i1} + 7) + \sum_{i=1} (4k_i p_i + k_i + p_i + 10) + k_s p_s - k_{11}p_{11} + 3k_n + k_{m1} + k - 5$$

<u>Proof</u>: We list below the number of basic computations performed

at each step of DBCA.

Step	Number of Basic Computations
A-1	$3k_{11}p_{11} + k_{11}$
A-2	$\sum_{i=2}^{m}(4k_{i1}p_{i1} + k_{i1} + 5)$
A-3	k_{m1}
B-1	k
B-2	$\sum_{i=1}^{s-1}(4k_ip_i + k_i + 5)$
C	$5(k_sp_s + k_s + 5)$
D	$\sum_{i=s+1}^{n}(4k_ip_i + k_i + 5)$
E	$3k_n - 1$
F	$\sum_{i=1}^{n}(p_i + 5)$
G	$\sum_{i=1}^{m}(p_{i1} + 2) + 1$

The addition of these values gives the result of the theorem.

Corollary 1: Suppose that all discretization levels equal k, in other words, $k_{i1} = p_{i1} = k_i = p_i = k$, for all admissable values of i. Then the computational complexity of DBCA is given by:

$$2(m^2 + n^2 + m + n)k^2 + (m^2 + n^2 + m + n + 5)k$$
$$+1/2(7m^2 + 9n^2 + 7m + 9n - 10)$$

Corollary 2: the DBCA has computational complexities of $O(k^2)$, $O(m^2)$, and $O(n^2)$ where k is the level of discretization of state and decision variables and m, n are numbers of stages in the diverging and main branches respectively.

Corollary 3: Let all three subbranches in the diverging branch system be approximately equal, i.e., $S = N/2 = M$. Then the complexity formula reduces to $2(5m^2 + 3m)k^2 + (5m^2 + 3m + 5)k + 1/2(43m^2 + 25m - 10)$.

Corollary 4: The computational complexity of DBCA is independent of the stage from where the diverging branch starts.

Proof: The formula for the complexity in theorem 2 does not contain the junction point s.

3. Converging Branch System

$$\begin{array}{ccccccc} & d_{m1} & & & & d_{11} & \\ x_{m1} & \boxed{m1} & x_{m-1,1} & \cdots & x_{11} & \boxed{11} & x_{01} \\ & r_{m1} & & & & r_{11} & \end{array}$$

$$\begin{array}{ccccccccccc} & d_n & & & & d_s & & & & d_1 & \\ x_n & \boxed{n} & x_{n-1} & \cdots\cdots & x_s & \boxed{s} & x_{s-1} & \cdots & x_1 & \boxed{1} & x_0 \\ & r_n & & & & r_s & & & & r_1 & \end{array}$$

The state and decision variables are given by:

$1 \leq x_i \leq k_i$, $1 \leq d_i \leq p_i$ where $1 \leq i \leq m$

$1 \leq x_{i1} \leq k_{i1}$, $1 \leq d_{i1} \leq p_{i1}$ where $1 \leq i \leq m$

The transformation and return tables are given by:

$x_{i1} = t_i(x_i,d_i)$, $r_i = r_i(x_i,d_i)$ where $i \leq i \leq n$

$x_{i-1,1} = t_{i1}(x_{i1},d_{i1}), r_i = r_{i1}(x_{i1},d_{i1})$ where $1 \leq i \leq m$, $i \neq s$

$x_{s-1} = t_s(x_s,x_{01},d_s), r_s = r_s(x_s,x_{01},d_s)$

$k = \max(k_{11},\ldots k_{m1},\ldots k_1,\ldots kn)$ and $k_d = k + 1$

3.1 High-level Algorithm For Converging Branch System

We describe a high-level algorithm to process the converging branch network. First, the branch is processed using initial-final state optimization of dynamic programming to obtain $f_{m1}(x_{m1},x_{m1})$, and this return is optimized over x_{m1} such that $1 \leq x_{m1} \leq k_{01}$. The optimal return is stored for later use. Stages 1 through s-1 of the main branch are processed using serial DP algorithm. The return at junction s and the optimal returns from s-1 and the branch are combined and then optimized over values of x_{01} and decision variable. Stages s+1 through n of the main branch are processed using serial DP algorithm. Finally,

the optimal return $f_n(x_n{*})$ is found by optimizing $f_n(x_n)$ over values of x_n. In the following, we present the details.

A. Initial-final state optimization of the branch

1. $f_{11}(x_{11},x_{01}) = \max \quad r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01}$;

2. For $2 \leq i \leq m$,

$$f_{i1}(x_{i1},x_{01}) = \max_{1 \leq d_{i1} \leq p_{i1}}(r_{i1}(x_{i1},d_{i1}) + f_{i-1,1}(t_{i-1}(x_{i1},d_{i1}),x_{01}));$$

B. Process the optimal return at stage m_1

$$f_{m1}(x_{01}) = \max_{1 \leq x_{m1} \leq k_{m1}}(f_{m1}(x_{m1},x_{01})),$$

Store $f_{m1}(x_{01})$;

C. Process stages 1 through s-1

1. $f_0(x_0) = 0$,

2. $f_i(x_i) = \max_{1 \leq d_i \leq p_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i))), \; 1 \leq i \leq s-1;$

Store $d_i{*}$ which optimizes the return.

D. Process junction s:

$$f_s(x_s) = \max_{\substack{1 \leq d_s \leq p_s \\ 1 \leq x_{01} \leq k_{01}}}(r_s(x_s,x_{01},d_s) + f_{s-1}(t_s(x_s,d_s) + f_{m1}(x_0))$$

Store $d_s{*}$ and $x_{01}{*}$ which optimize the return.

E. Process stages s + 1 through n:

1. For $s + 1 \leq i \leq n$,

$$f_i(x_i) = \max_{1 \leq d_i \leq p_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)));$$

Store $d_i{*}$ which optimizes f_i.

F. Optimal return $f_n(x_n^*)$ at stage n:

$$f_n(x_n^*) = \max_{1 \leq x_n \leq k_n} f_n(x_n)$$

G. Optimal decisions for stages n through 1:

1. For i=n down to s+1;

   Find d_i^*;

   $x_{i-1}^* = t_i(x_i^*, d_i^*)$;

2. At junction s:

   Find d_s^*;

   $x_{s-1}^* = t_s(x_s^*, x_{01}^*, d_s^*)$;

3. For i = s-1 down to 1:

   Find d_i^*;

   $x_{i-1}^* = t_i(x_i^*, d_i^*)$;

H. Optimal decisions for stages 11 through m_1 of the branch:

1. $f_{11}(x_{11}) = \max_{1 \leq d_{11} \leq p_{11}} (r_{11}(x_{11}, d_{11}) : (t_{11}(x_{11}, d_{11}) = x_{01}^*)$;

2. For $2 \leq i \leq m$:

   $f_{i1}(x_{i1}) = \max_{1 \leq d_i \leq p_i} (r_{i1}(x_{i1}, d_{i1}) + f_{i-1,1}(t_{i1}(x_{i1}, d_{i1})))$;

   Store d_{i1}^*;

3. Find $\max_{1 \leq x_{m1} \leq k_{m1}} (f_{m1}(x_{m1}))$;

   Let x_{m1}^* optimize f_{m1};

4. For i = m down to 1:

   Find d_{i1}^*;

   $x_{i-1,1} = t_{i1}(x_{i1}^*, d_{i1}^*)$;

3.2 Converging Branch Computing Algorithm (CBCA)

We follow the notations of section 2.1, for processing converging branch system, tables F1(0:1,1:k,1:k) and F(0:1,1:k) are needed to store converging branch and main branch optimal returns. A table FM(1:k) is needed to store the optimal return at the stage m1 for later use. Now, we describe the CBCA in detail.

A. (* stages 11 through m1 *)

1. (* Initialize the table F1(1,.,.) by processing stage 11 *)

For x_{11} = 1 to k_{11} do

For x_{01} = 1 to k_{01} do

$F1(1,x_{11},x_{01}) = r_{11}(x_{11},d_{11}^*) = \max(r_{11}(x_{11},d_{11}) \mid t_{11}(x_{11},d_{11}) = x_{01});$

2. (* Process stages 21 through m1, using tables F1(0,.,.) and F1(1,.,.)*)

For i=2 to m do

st = (i-1)mod 2;

dt = i mod 2;

For x_{i1} = 1 to k_{i1} do

For x_{01} = 1 to k_{01} do

$F1(dt,x_{i1},x_{01}) = r_{i1}(x_{i1},d_{i1}) + F1(st,t_{i1}(x_{i1},d_{i1}),x_{01})$

B. (* Maximize $f_{m1}(x_{m1},x_{01})$ over values of x_{m1}, and store the optimal values in table FM(.) *)

st = m mod 2;

For x_{01} = 1 to k_{01} do

$$FM(x_{01}) = F1(st,x_{m1},x_{01}) = \max_{1 \le x_{m1} \le k_{m1}} F1(st,x_{m1},x_{01});$$

C. (* stages 1 through s-1 *)

1. (* Initialize by putting 0's in the source table *)

```
For i=1 to k do
     F(0,i) = 0;
```

2.

```
For i=1 to s-1 do
     st = (i-1) mod 2;
     dt = i mod 2;
     For x_i=1 to k_i do
```

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i^*)) = \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d;$$

D. (* At junction there are two source tables, FM(.) and F(st,.) *)

```
st = (s-1) mod 2;
dt = s mod 2;
For x_s=1 to k_s do
```

$$F(dt,x_s) = r_s(x_s,x_{01}^*,d_s^*) + F(st,t_s(x_s,x_{01}^*,d_s^*)) + FM(x_{01}));$$

$$= \max_{1 \le d_s \le p_s}(r_s(x_s,x_{01}^*,d_s) +$$

$$F(st,t_s(x_s,x_{01}^*,d_s^*) = t_s(x_s,x_{01}^*,d_s^*) + k_($$

E. (* stages s+1 through n *)

```
For i = s+1 to n do
     st = (i-1) mod 2;
     dt = i mod 2;
     For x_i=1 to k_i do
```

$$F(dt,x_i) = r_i(x_i,d_i) + F(st,t^i(x_i,d_i^*)$$

$$= \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d;$$

F. (* Find optimal return $f_n(x_n^*)$ *)

$$F_n(x_n^*) = F(st,x_n^*) = \max_{1 \le x_n \le k_n} F(st,x_n);$$

G. (* Optimal decisions, stages n through 1 *)

1. For i=n down to s+1 do

   Search x_i^*-row of t_i for $t_i(x_i^*,d_i^*) \ge k_d$ for $1 \le d_i \le p_i$;

   $x_{i-1}^* = t_i(x_i^*,d_i^*) \bmod k_d$;

2. Search $x_s^* x_{01}^*$-row of t_s for $t_s(x_s^*,x_{01}^*,d_s^*) \ge k_d$, $1 \le d_s \le p_s$;

   $x_{s-1}^* = t_s(x_s^*,x_{01}^*,d_s^*) \bmod k_d$;

3. For i = s-1 down to 1 do

   Search x_i^*-row for $t_i(x_i^*,d_i^*) \ge k_d$, $1 \le d_i \le p_i$;

   $x_{i-1}^* = t_i(x_i^*,d_i^*) \bmod k_d$;

H. (* Optimal decisions, stages 11 through m_1 *)

1. For x_{11}=1 to k_{11} do

   $$F1(1,x_{11},x_{01}^*) = r_1(x_{11},d_{11}^*) = \max_{1 \le d_{11} \le p_{11}} (r_1(x_{11},d_{11})|$$

   $$t_{11}(x_{11},d_{11}^*) = t_{11}(x_{11},d_{11}^*) + k_d$$

2. For i=2 to m do

   st = (i-1) mod 2;

   dt = i mod 2;

   For x_{i1}=1 to k_{i1} do

   $$F1(dt,x_{i1},x_{01}^*) = r_{i1}(x_{i1},d_{i1}^*) + F1(st,t_{i1}(x_{i1},d_{i1}^*)x_0$$

   $$= \max_{1 \le d_{i1} \le p_{i1}} (r_{i1}(x_{i1},d_{i1}^*) + F1(st,t_i(x_{i1},d_{i1})x_{01}^*));$$

   $$t_{i1}(x_{i1},d_{i1}^*) = t_{i1}(x_{i1},d_{i1}^*) + k_d;$$

3. st = m mod 2;

$$F1(st, x_{m1}^*, x_{01}^*) = \max_{1 \le x_{m1} \le k_{m1}} F1(st, x_{m1}, x_{01}^*);$$

4. For i=m down to 1 do

   Search x_{i1}^*-row of t_i for $t_{i1}(x_{i1}^*, d_{i1}^*) \geq k_d$

   $1 \leq d_{i1}^*, \leq p_{i1}$;

   $x_{i-1}^*{}_{,1} = t_{i1}(x_{i1}^*, d_{i1}^*) \bmod k_d$;

3.3 Storage Complexity of CBCA

Here we use the same notations and techniques as in section 2.

Theorem 3: The storage complexity of CBCA is given by $2k^2 + k$ or $O(k^2)$.

Proof: The storage tables used are F1(0:1,1:K,1:K), F(0:1,1:K) and FM(1:K). However, when optimal returns from the branch have been transferred to FM, F1 can be released and the smaller table F may be used. So at any one time the maximum storage used will be no more than $2k^2 + k$.

3.4 Computational Complexity of CBCA

The notations and techinques of section 2 are used here also.

Theorem 4: The computational complexity of CBCA is given by

$$\sum_{i-1}^{m} (4k_{i1}k_{01}p_{i1} + 4k_{i1}p_{i1} - k_{i1}k_{01} + k_{i1} + p_{i1} + 12$$

$$+ \sum_{i=1}^{n} (4k_i p_i + k_i + p_i + 7) + 5k_s k_{01} p_s + 3k_{m1}k_{01} - 4k_s p_s - k_{01}$$

$$+ 3k_n + 3k_{m1} + k - 8$$

Proof: The number of computations performed at each step of CBCA is listed below:

Step	Number of Basic Operations
A.1	$4k_{i1}k_{01}p_{i1} - k_{i1}k_{01}$
A.2	$\overset{m}{\underset{i=2}{}}(4k_{i1}k_{01}p_{i1} - k_{i1}k_{01} + 5)$
B	$3k_{m1}k_{01} - k_{01} + 2$
C.1	k
C.2	$\overset{s-1}{\underset{i=1}{}}(4k_ip_i + k_i + 5)$
D	$5k_sk_{01}p_s + k_s + 5$
E	$\overset{n}{\underset{i=s+1}{}}(4kipi + k_i + 5)$
F	$3k_n + 1$
G.1	$\overset{n}{\underset{i=s+1}{}}(p_{i1} + 2)$
G.2	$p^s + 2$
G.3	$\overset{s-1}{\underset{i=1}{}}(p_i + 2)$
H.1	$4k_{11}p_{11} + k_{11}$
H.2	$\overset{m}{\underset{i=2}{}}(4k_{i1}p_{i1} + k_{i1} + 5)$
H.3	$3k_{m1} + 1$
H.4	$\overset{m}{\underset{i=1}{}}(p_{i1} + 2)$

Addition of these results gives the complexity stated in the theorem.

Corollary 5: Let $k_{i1} = p_{i1} = k_i = p_i = k$ for all admissible values f_i. Then the computational complexity formula for CBCA reduces to

$$(2m^2 + 2m + 5)k^3 + 1/2(m^2 + 4n^2 + m + 4n - 2)k^2 + (m^2 + n^2 + m + n + 6)k + 1/2(12m^2 + 7n^2 + 12m + 7n - 16)$$

Corollary 6: For the conditions stated in Corollary 5, the CBCA has complexities $O(k^3)$, $O(m^2)$, $O(n^2)$.

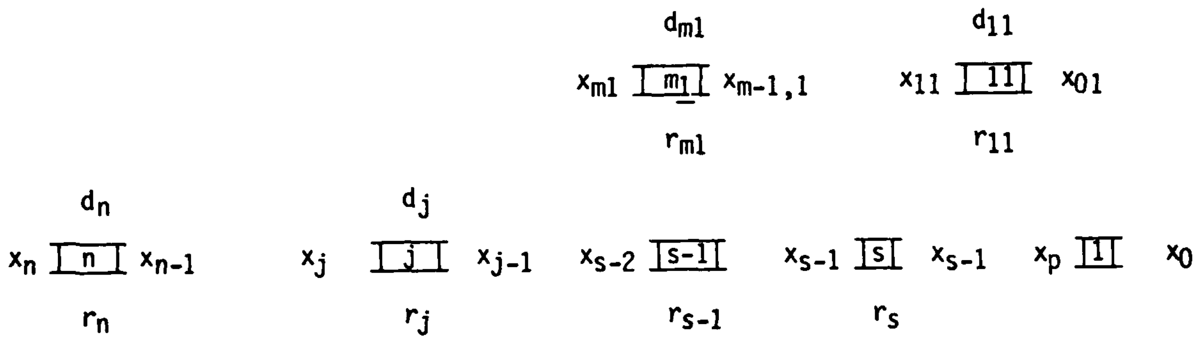
Corollary 7: Assuming that the numbers of stages in the three subbranches of the converging branch network are approximately equal ($s=n/2=m$), the formula for the complexity reduces to

$$(2m^2 + 2m + 5)k^3 + 1/2(m^2 + 4n \quad + m + 4n - 2)k^2 + (5m^2 + 3m + 6)k + (20m^2 + 13m - 8)$$

Corollary 8: The computational complexity of CBCA is independent of the junction stage.

Proof: The result of theorem 5 does not contain any s.

4. Feedforward Loop System



The state and decision variables are given by

$1 \leq x_i \leq k_i$, $1 \leq d_i \leq p_i$ where $1 \geq i \leq n$;

$1 \leq x_{i1} \leq k_{i1}$, $1 \leq d_{i1} \leq p_{i1}$ where $1 \leq i \leq m$;

$k = \max(k_{11},\ldots k_{m1},\ldots,k_1\ldots,k_n)$; $k_d = k + 1$

Transformations and returns are described by tables

$x_{m1} = t_{j1}(x_j,d_j)$;

$x_{i-1,1} = t_{i1}(x_{i1},d_{i1})$, $1 \leq i \leq m$;

$x_{s-1} = t_s(x_s,x_{01},d_s)$;

$x_{i-1} = t_i(x_i,d_i)$, $1 \leq i \leq n$, $i \neq s$;

$r_s = r_s(x_s,x_{01},d_s)$;

$r_i = r_i(x_i,d_i)$, $1 \leq i \leq n$, $i \neq s$;

$r_{i1} = r_{i1}(x_{i1},d_{i1})$, $1 \leq i \leq m$.

4.1 High-level Algorithm for Feedforward Loop System

First, the loop-branch is processed as an initial-final state optimization using serial DP-algorithm, and the optimal return $f_{m1}(x_{m1},x_{01})$ is stored. Similarly, using serial DP-algorithm stages 1 through s-1 are processed and $f_{s-1}(x_{s-1})$ is stored. Stages s through j-1 are processed obtaining returns which depend on x_{01}. The branch

optimal return and that from the stage j-1 are combined with the return at j^1th stage, and the resultant return is optimized over values of x_{01} and d_j. Stages j+1 through n are handled using serial DP-algorithm, and optimal return $f_n(x_n^*)$ is determined.

The details of this algorithm are given below:

A. For branch including stages 11 through m_1, perform initial-final state optimization.

1. $f_{11}(x_{11},x_{01}) = \max\limits_{d_{11}}(r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01});$

2. $f_{i1}(x_{i1},x_{01}) = \max\limits_{d_{i1}}(r_{i1}(x_{i1},d_{i1}) + f_{i-1}(t_{i1}(x_{i1},d_{i1}),x_{01})),$

$$2 \leq i \leq m;$$

Store $f_{m1}(x_{m1},x_{01});$

B. Serial DP-algorithm for stages 1 through s-1:

1. $f_0(x_0) = 0$ (* initialize *)

2. $f_i(x_i) = \max\limits_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i))$

   Store optimal decision d_i^* which optimizes f_i.

3. Store $f_{s-1}(x_{s-1})$.

C. At the stage s perform usual DP optimization.

$$f_s(x_s,x_{01}) = \max_{d_s}(r_s(x_s,x_{01},d_s) + f_{s-1}(t_s(x_s,x_{01},d_s)));$$

D. Stages s+1 through j-1:

$$f_i(x_i,x_{01}) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i),x_{01}));$$

E. At j combine all returns:

$$f_j(x_j) = \max_{x_{01},d_j}(r_j(x_j,d_j) + f_{j-1}(t_j(x_j,d_j),x_{01}) + f_{m1}(t_{ji}(x_j,d_j),x_{01}));$$

Store the optimal decision d_j^* and x_{01}^*.

F. Stages j+1 through n:

$$f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i));$$

Store the optimal decision d_i*.

G. $$f_n(x_n^*) = \max_{x_n} f_n(x_n)$$

H. Optimal decision stages n through 1:

1. $$f_s(x_s) = \max_{d_s}(r_s(x_s,x_{01}^*,d_s) + f_{s-1}(t_s(x_s,x_{01}^*,d_s)));$$

   Store the optimal return at d_s*.

2. For $s+1 \le i \le j-1$:

   $$f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)));$$

   Store d_i*.

3. For i=n down to s+1:

   Find d_i*,

   $x_{i-1}^* = t_i(x_i^*,d_i^*)$,

   If i=j then $XJ=x_j^*$.

4. Find d_s*;

   $x_{s-1}^* = t_s(x_s^*,x_{01}^*,d_s^*)$;

5. For i=s down to 1:

   Find d_i*;

   $x_{i-1}^* = t_i(x_i^*,d_i^*)$;

I. Optimal decision for stages 11 through m_1:

1. $$f_{11}(x_{11}) = \max_{d_{11}}(r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01}^*);$$

2. For $2 \leq i \leq m$:

$f_{i1}(x_{i1}) = \max(r_{i1}(x_{i1},d_{i1}) + f_{i-1}(t_{i1}\ (x_{i1},d_{i1})))$;

Store optimal decision $d_{i1}*$;

3. $x_{m1}* = t_{ji}(XJ,d_j*)$;

4. For i=m down to 1:

Find $d_{ij}*$;

$x_{i-1}{}^{*}{}_{,1} = t_{i1}(x_{i1}*,d_{i1}*)$;

4.2 Feedforward Loop Computer Algorithm (FFLCA)

In FFLCA, tables F1(0:1,1:K,1:K) and F(0:1,1:K) are needed for storing optimal returns. Moreover, FS(1:K) and FM(1:K,1:K) are also used for storage of $f_{s-1}(x_{s-1})$ and $f_{m1}(x_{m1},x_{01})$ for further processing.

A. (* Branch stages 11 through m_1 *)

1. (* Initialize F1(1,.,.) by processing stage 11 *)

For x_{11}=1 to k_{11} do

For x_{01} to k_{01} do

$$F1(1,x_{11},x_{01}) = r_{11}(x_{11},d_{11}*) = \max_{1 \leq d_{11} \leq p_{11}}(r_{11}(x_{11},d_{11})|$$

$$t_{11}(x_{11},d_{11}) = x_{01});$$

2. For i=2 to m do

st = (i-1) mod 2;

dt = i mod 2;

For x_{i1}=1 to k_i, do

For x_{01}=1 to k_{01} do

$$F1(dt,x_{i1},x_{01}) = r_{i1}(x_{i1},d_{i1}*) + F1(st,t_{i1},(x_{i1},d_{i1}*),x_0);$$

$$= \max_{1 \leq d_{i1} \leq p_{i1}}(r_{i1}(x_{i1},d_{i1}) + F1(st,t_{i1}(x_{i1},d_{i1}*),x_0));$$

3. (* Store the mth stage results into FM(.,.) for later use *)

For x_{m1}=1 to k_{m1} do

For x_{01}=1 to k_{01} do

$FM(x_{m1}, x_{01}) = F1(dt, x_{i1}, x_{01});$

B. (* Stage 1 through s-1 *)

1. (* Initialize by putting 0's in the source table *)

For i=1 to k do

F(0,i) = 0;

2. For i=1 to s-1 do

st = (i-1) mod 2;

dt = i mod 2;

For x_i=1 to k_i do

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i^*))$$

$$= \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d;$$

3. For x_{s-1}=1 to k_{s-1} do

$FS(x_{s-1}) = F(dt,x_{s-1});$

C. (* stage s *)

st = (s-1) mod 2;

dt = s mod 2;

For x_s=1 to k_s do

For x_{01}=1 to k_{01} do

$$F1(dt,x_s,x_{01}) = r_s(x_s,x_{01},d_s^*) + FS(t_s(x_s,x_{01},d_s^*))$$

$$= \max_{1 \le d_s \le p_s}(r_s(x_s,x_{01},d_s) + FS(t_s(x_s,x_{01},d_s^*));$$

D. (* Stages s+1 through j-1 *)

For i= s+1 to j-1 do

st = (i-1) mod 2;

dt = i mod 2;

For x_i=1 to k_i do

For x_{01}=1 to k_{01} do

$$F1(dt,x_i,x_{01}) = r_i(x_i,d_i^*) + F1(st,t_i(x_i,d_i^*),x_{01}) = \max_{1 \leq d_i \leq p_i}(r_i(x_i,d_i) + F1(st,t_i(x_i,d_i),x_{01}));$$

E. (* Stage j *)

st = (j-1) mod 2;

dt = j mod 2;

For x_j=1 to k_j do

$$F(dt,x_j) = r_j(x_j,d_j^*) + F1(st,t_j(x_j,d_j^*),x_{01}^*) + FM(t_{ji}(x_j,d_j),x_{01}));$$

$$= \max_{\substack{1 \leq x_{01} \leq k_{01} \\ 1 \leq d_j \leq p_j}}(r_j(x_j,d_j^*) + F1(st,t_j(x_j,d_j),x_{01}) + FM(t_{ji}(x_j,d_j),x_0$$

$t_j(x_j,d_j^*) = t_j(x_j,d_j^*) + k_d$; $t_{j1}(x_j,d_j^*) = t_{j1}(x_j,d_j^*) + k_d$;

F. (* Stages j+1 through n *)

For i=j+1 to n do

st = (i-1) mod 2;

dt = i mod 2;

For x_1=1 to k_i do

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i^*)) = \max_{1 \leq d_i \leq p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i))$$

$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d$

G. (* Find maximal return $f_n(x_n^*)$ *)

$$F_n = F(dt, x_n^*) = \max_{1 \le x_n \le k_n} F(dt, x_n);$$

H. (* Optimal decisions for stages n through 1 *)

1. $dt = s \bmod 2;$

   For x_s=1 to k_s do

   $$F(dt, x_s, x_{01}^*) = r_s(x_s, x_{01}^*, d_s^*) + FS(t_s(x_s, x_{01}^*, d_s^*))$$

   $$= \max_{1 \le d_s \le p_s}(r_s(x_s, x_{01}^*, d_s) + FS(t_s(x_s, x_{01}^*, d_s)));$$

   $$t_s(x_s, x_{01}^*, d_s^*) = t_s(x_s, x_{01}^*, d_s^*) + k_d$$

2. For i = s+1 to j-1 do

   $st = (i-1) \bmod 2;$

   $dt = i \bmod 2;$

   For x_i=1 to k_i do

   $$F1(dt, x_i, x_{01}^*) = r_i(x_i, d_i^*) + F1(st, t_i(x_i, d_i^*), x_{01}^*)$$

   $$= \max_{1 \le d_i \le p_i}(r_i(x_i, d_i) + F(st, t_i(x_i, d_i), x_{01}^*));$$

   $$t_i(x_i, d_i^*) = t_i(x_i, d_i^*) + k_d;$$

3. For i=n down to s+1 do

   Search x_i^*-row of t_i for $t_i(x_i^*, d_i^*) \ge k_d$, $1 \le d_i \le p_i$;

   If i=j, then $x_J = x_j^*$;

   $$x_{i-1}^* = t_i(x_i^*, d_i^*) \bmod k_d;$$

4. Search x_s^*-row of t_s for $t_s(x_s^*, x_{01}^*, d_s^*) \ge k_d$, $1 \le d_s \le p_s$;

   $x_{s-1}^* = t_s(x_s^*, x_{01}^*, d_s^*) \bmod k_d;$

5. For i = s-1 down to 1 do

   Search x_i-row of t_i for $t_i(x_i^*, d_i^*) \ge k_d$, $1 \le d_i \le p_i$;

   $x_{i-1}^* = t_i(x_i^*, d_i^*) \bmod k_d;$

I. (* Optimal decisions for stages 11 through m_1 *)

1. For x_{11}=1 to k_{11} do

$$F1(1,x_{11},x_{01}^*) = r_{11}(x_{11},d_{11}^*) = \max_{1 \le d_{11} \le p_{11}}(r_{11}(x_{11},d_{11}) | t_{11}(x_{11},d_{11})=x_{01}^*);$$

$$t_{11}(x_{11},d_{11}^*) = t_{11}(x_{11},d_{11}^*) + k_d;$$

2. For i=2 to m do

st = (i-1) mod 2;

dt = i mod 2;

For x_{i1}=1 to k_{i1} do

$$F1(dt,x_{i1},x_{01}^*) = r_{i1}(x_{i1},d_{i1}^*) + F1(st,t_{i1}(x_{i1},d_{i1}^*),x_{01}^*)$$

$$= \max_{1 \le d_{i1} \le p_{i1}}(r_{i1}(x_{i1},d_{i1}) + F1(st,t_{i1}(x_{i1},d_{i1}),x_{01}^*));$$

$$t_{i1}(x_{i1},d_{i1}^*) = t_i(x_{i1},d_{i1}^*) + k_d;$$

3. Search x_J-row of t_{ji} for $t_{ji}(x_J,d_j^*) \ge k_d$, $1 \le d_j \le p_j$;

$x_{m1}^* = t_{j1}(x_J,d_j^*) \bmod k_d$;

4. For i=m down to 1 do

Search x_{i1}^*-row of t_{i1} for $t_{i1}(x_{i1}^*,d_{i1}^*) \ge k_d$, $1 \le d_{i1} \le p_{i1}$;

$x_{i-1,1}^* = t_{i1}(x_{i1}^*,d_{i1}^*) \bmod k_d$;

4.2 Storage Complexity of FFLCA

We use the same notations and techniques as in section 2.

Theorem 5: The storage complexity of FFLCA is given by $2k^2 + 2k$ or $O(k^2)$.

Proof: Two tables F1(0,1:k,1:k) each with k^2 storage requirement are used to process branch. One of these tables is retained to store $f_{m1}(x_{m1},x_{01})$ for later use while the space taken by the other can be released. Tables F(0:1,1:k) using 2k spaces are used to process main branch. Moreover, FS(1:k) and FM(1:k,1:k) are used for storage of optimal returns for later use. Hence,

at one time no more than $2k^2 + 2k^2$ spaces are required.

4.3 Computational Complexity of FFLCA

The notations and techniques of section 2 are also used here.

Theorem 6: The computational complexity of FFLCA is given by

$$\sum_{i=1}^{m}(4k_{i1}k_{01}p_{i1} + 4k_{i1}p_{i1} - k_{i1}k_{01} + k_{i1} + p_{i1} + 12) \quad \sum_{i=s}^{j}(4k_ik_{01} - k_ik_{01} + 5)$$

$$\sum_{i=1}^{n}(4k_ip_i + k_i + p_i + 7) + 2k;$$

$$+ k_jk_{01}p_j + k_jk_{01} - 4k_jp_j + k_{m1}k_{01} + 3k_n + k_s + p_j + k + (n^2 - s^2 + n - s - 17).$$

Proof: The number of computations performed at each step is listed below.

Step	Number of Computations
A.1	$4k_{11}k_{01}p_{11} - k_{11}k_{01}$
A.2	$\sum_{i=2}^{m}(4k_{i1}k_{01}p_{i1} - k_{i1}k_{01} + 5)$
A.3	$k_{m1}k_{01}$
B.1	k
B.2	$\sum_{i=1}^{s-1}(4k_ip_i + k_i + 5)$
B.3	k_{s-1}
C	$4k_sk_{01}p_s - k_sk_{01} + 5$
D	$\sum_{i=s+1}^{j-1}(4k_ik_{01}p_i - k_ik_{01} + 5)$
E	$5k_jk_{01}p_j + 3k_j + 5$
F	$\sum_{i=j+1}^{n}(4k_ip_i + k_i + 5)$
G	$3k_n - 1$
H.1	$4k_sp_s + k_s + 2$

H.2	$\sum_{i=s+1}^{j-1}(4k_i p_i + k_i + 5)$
H.3	$\sum_{i=s+1}^{n}(p_i + 4)$
H.4	$p_s + 2$
H.5	$\sum_{i=1}^{s-1}(p_i + 2)$
I.1	$4k_{11}p_{11} + k_{11}$
I.2	$\sum_{i=2}^{m}(4k_{i1}p_{i1} + k_{i1} + 5)$
I.3	$p_j + 2$
I.4	$\sum_{i=1}^{m}(p_{i1} + 2)$

Addition of all these gives the result of the theorem.

Corollary 8: Suppose that $k_{i1} = p_{i1} = k_i = p_i = k$ for all admissible values of i. Then the formula for computational complexity for FFLCA reduces to

$$2(m^2 + j^2 - s^2 + m + j + s + 1/2)k^3 + 1/2(3m^2 + 4n^2 + s^2 - j^2 + 3m + 4n - j - s - 4$$
$$+ (m^2 + n^2 + m + n + 8)k + 1/2(12m^2 + n^2 + sj^2 - 6s^2 + 5j + 4s + n + 12m - 34)$$

In other words it has representations $O(m^2)$, $O(j^2-s^2)$, $O(n^2)$, and $O(k^3)$.

Corollary 9: Assume that all three subbranches in feedforward loop system are approximately equal , i.e., $s = j/2 = n/3 = m$. The formula for computational complexity for FFLCA becomes:

$$(8m^2 + 8m + 1)k^3 + 2(9m^2 + 3m - 1)k^2 + 2(5m^2 + 2m + 4)k + 1/2(35m^2 + 29m - 34).$$

Corollary 10: The computational complexity of FFLCA depends on the junction points j and s with complexity described as $O(j^2-s^2)$.

H.2	$\sum_{i=s+1}^{j-1}(4k_ip_i + k_i + 5)$
H.3	$\sum_{i=s+1}^{n}(p_i + 4)$
H.4	$p_s + 2$
H.5	$\sum_{i=1}^{s-1}(p_i + 2)$
I.1	$4k_{11}p_{11} + k_{11}$
I.2	$\sum_{i=2}^{m}(4k_{i1}p_{i1} + k_{i1} + 5)$
I.3	$p_j + 2$
I.4	$\sum_{i=1}^{m}(p_{i1} + 2)$

Addition of all these gives the result of the theorem.

Corollary 8: Suppose that $k_{i1} = p_{i1} = k_i = p_i = k$ for all admissible values of i. Then the formula for computational complexity for FFLCA reduces to

$$2(m^2 + j^2 - s^2 + m + j + s + 1/2)k^3 + 1/2(3m^2 + 4n^2 + s^2 - j^2 + 3m + 4n - j - s - 4$$
$$+ (m^2 + n^2 + m + n + 8)k + 1/2(12m^2 + n^2 + sj^2 - 6s^2 + 5j + 4s + n + 12m - 34)$$

In other words it has representations $O(m^2)$, $O(j^2-s^2)$, $O(n^2)$, and $O(k^3)$.

Corollary 9: Assume that all three subbranches in feedforward loop system are approximately equal , i.e., $s = j/2 = n/3 = m$. The formula for computational complexity for FFLCA becomes:

$$(8m^2 + 8m + 1)k^3 + 2(9m^2 + 3m - 1)k^2 + 2(5m^2 + 2m + 4)k + 1/2(35m^2 + 29m - 34).$$

Corollary 10: The computational complexity of FFLCA depends on the junction points j and s with complexity described as $O(j^2-s^2)$.

5. Feedback Loop System

$$\begin{array}{ccc} & d_{11} & \\ x_{01} & \boxed{11} & x_{11} \\ & r_{11} & \end{array} \qquad \begin{array}{ccc} & d_{m1} & \\ x_{m-,1} & \boxed{m1} & x_{m1} \\ & r_{m1} & \end{array}$$

$$\begin{array}{ccc} & d_n & \\ x_n & \boxed{n} & x_{n-1} \\ & r_n & \end{array} \qquad \begin{array}{ccc} & d_j & \\ x_j & \boxed{j} & x_{j-1} \\ & r_j & \end{array} \qquad \begin{array}{ccc} & d_s & \\ x_s & \boxed{s} & x_{s-1} \\ & r_s & \end{array} \qquad \begin{array}{ccc} & d_1 & \\ x_1 & \boxed{1} & x_0 \\ & r_1 & \end{array}$$

The state and decision variables are given by:

$1 \leq x_i \leq k_i$, $1 \leq d_i \leq p_i$ $\quad 1 \leq i \leq n$;

$1 \leq x_{i1} \leq k_{i1}$, $1 \leq d_{i1} \leq p_{i1}$ $\quad 1 \leq i \leq m$;

$k = \max k_{i1}, \ldots k_{m1}, k_i, \ldots k_n$ and $k_d = k + 1$;

The states and returns are given by the tables:

$x_{j-1} = t_j(x_j, x_{01}, d_j)$; $r_j = r_j(x_j, x_{01}, d_j)$;

$x_{i-1} = t_i(x_i, d_i)$; $r_i = r_i(x_i, d_i)$, $1 \leq i \leq n$, $i \neq j$.

$x_{m1} = t_{s1}(x_s, d_s)$;

$x_{i-1,1} = t_{i1}(x_{i1}, d_{i1})$; $r_{i1} = r_{i1}(x_{i1}, d_{i1})$, $1 \leq i \leq m$.

5.1 High-Level Feedback Loop Algorithm

Process the branch including stages 11 through m_1 using initial-final state DP-algorithm, and store the optimal return $f_{m1}(x_{m1}, x_{01})$ for later use. Process stages 1 through s-1 using serial DP-algorithm, and combine this and the branch optimal returns with the return at the stage s. Optimize the result over values of d_s to obtain $f_s(x_s, x_{01})$. Process stages s+1 through j-1 using serial DP-algorithm. After combining the

optimal return from the stage j-1 and the return at j, optimize the result over values of x_{01} and d_j to obtain $f_j(x_j)$. Process stages j+1 through n using serial DP-algorithm and find the optimal return by optimizing $f_n(x_n)$ over values of x_n.

A. Stages 11 through m_1

1. $f_{11}(x_{11},x_{01}) = \max_{d_{11}} r_{11}(x_{11},d_{11}) | t_{11}(x_{11},d_{11}) = x_{01}$;

2. For $2 \leq i \leq m$:

   $f_{i1}(x_{i1},x_{01}) = \max_{d_{i1}}(r_{i1}(x_{i1},d_{i1}) + f_{i-1,1}(t_{i1}(x_{i1},d_{i1}),x_{01}))$;

3. Store $f_{m1}(x_{m1},x_{01})$ for later use.

B. Stages 1 through s-1:

1. $f_0(x_0) = 0$;

2. For $1 \leq i \leq s-1$:

   $f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)))$;

   store the optimal decision d_i^*;

3. Store $f_{s-1}(x_{s-1})$;

C. At junction s:

$$f_s(x_s,x_{01}) = \max_{d_s}(r_s(x_s,d_s) + f_{s-1}(t_s(x_s,d_s)) + f_{m1}(t_{s1}(x_s,d_s),x_{01})));$$

D. Stages s+1 through j-1:

$$f_i(x_i,x_{01}) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i),x_{01}));$$

E. Junction j:

$$f_j(x_j) = \max_{d_j,x_{01}}(r_j(x_j,x_{01},d_j) + f_{j-1}(t_j(x_j,x_{01},d_j),x_{01}));$$

Store d_j^* and x_{01}^* for later use.

F. Stages j+1 through n:

$$f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i));$$

Store the optimal decision d_i*;

G. Find the optimal return $f_n(x_n^*)$:

$$f_n(x_n^*) = \max_{x_n} f_n(x_n);$$

H. Optimal decisions for stage n through 1:

1. $f_s(x_s) = \max_{d_s}(r_s(x_s,d_s) + f_{s-1}(t_s(x_s,d_s) + f_{m1}(t_{s1}(x_s,d_s),x_{01}^*));$

   Store the optimal decision d_s*;

2. For $s+1 \leq i \leq j-1$:

   $f_i(x_i) = \max_{d_i}(r_i(x_i,d_i) + f_{i-1}(t_i(x_i,d_i)));$

   Store the optimal decision d_i*;

3. For i=n down to j+1:

   Find d_i*;

   $x_{i-1}^* = t_i(x_i^*,d_i^*);$

4. Find d_j*;

   $x_{j-1}^* = t_j(x_j^*,x_{01}^*,d_j^*);$

5. For i = j-1 down to 1:

   Find d_i*;

   $x_{i-1}^* = t_i(x_i^*,d_i^*);$

   If i = s+1 then XS = x_s* and DS = d_s*.

I. For optimal decisions, stages m_1 through 11:

1. $f_{11}(x_{11}) = \max_{d_{11}}(r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01}^*);$

   Store the optimal decision d_{11}*;

2. For $2 \leq i \leq m$:

$$f_{i1}(x_{i1}) = \max_{d_{i1}}(r_{i1}(x_{i1},d_{i1}) + f_{i-1,1}(t_{i1}(x_{i1},d_{i1})));$$

Store d_{i1}^*;

3. Find $x_{m1}^* = t_{s1}(x_s,d_s)$;

4. For i=m down to 1:

   Find d_{m1}^*;

   $x_{i-1,1}^* = t_{i1}(x_{i1}^*,d_{i1}^*)$;

5.2 Feedback Loop Computer Algorithm (FBLCA)

The storage requirements for FBLCA are tables F(0:1,1:k), F1(0:1,1:k,1:k) for optimal returns and FS(1:k) and FM(1:k,1:k) for storage of $f_{s-1}(x_{s-1})$ and $f_{m1}(x_{m1},x_{01})$ for later use. The notation and techniques used here are the same as in section 2.

A. (* stages 11 through m_1 *)

1. (* Initialize F1(1,.,.) by processing stage 11 *)

   For x_{11}=1 to k_{11} do

   For x_{01}=1 to k_{01} do

$$F1(1,x_{11},x_{01}) = r_1(x_{11},d_{11}^*) = \max_{1 \leq d_{11} \leq p_{11}}(r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01});$$

2. For i=2 to m do

   st = (i-1) mod 2;

   dt = i mod 2;

   For x_{i1} to k_{i1} do

   For x_{01}=1 to k_{i1} do

$$F1(dt,x_{i1},x_{01}) = r_{i1}(x_{i1},d_{i1}) + F1(st,t_{i1}(x_{i1},d_{i1}^*),x_{01}));$$

$$= \max_{1 \leq d_{i1} \leq p_{i1}}(r_{i1}(x_{i1},d_{i1}) + F1(st,t_{i1}(x_{i1},d_{i1}),x_{01}));$$

3. (* store the mth stage results into FM(.,.) *)

For x_m=1 to k_{m1} do

For x_{01}=1 to k_{01} do

$FM(x_{m1},x_{01}) = F1(dt,x_{i1},x_{01});$

B. (* stages 1 through s-1 *)

1. (* Initialize by putting 0's in the source table *)

For i=1 to k do

F(0,i) = 0

2. For i=1 to s-1 do

st = i mod 2;

dt = i mod 2;

For x_i=1 to k_i do

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i));$$

$$= \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

$$t_i(x_i,d_i^*) = t_i(x_i^*,d_i^*) + k_d;$$

3. For x_{s-1}=1 to k_{s-1} do

$FS(x_{s-1}) = F(dt,x_{s-1});$

C. (* Process s *)

st = (s-1) mod 2;

dt = s mod 2;

For x_s=1 to k_s do

For x_{01}=1 to k_{01} do

$$F1(dt,x_s,x_{01}) = r_s(x_s,d_s^*) + FS(t_s(x_s,d_s^*) + FM(t_{s1}(x_s,d_s^*),x_{01});$$

$$= \max_{1 \le d_s \le p_s}(r_s(x_s,d_s) + FS(t_s(x_s,d_s)) + FM(t_{s1}(x_s,d_s)x_{01}));$$

D. (* stages s+1 to j-1 *)

For i = s+1 to j-1 do

st = (i-1) mod 2;

dt = i mod 2;

For x_i=1 to k_i do

For x_{01}=1 to k_{01} do

$$F1(dt,x_i,x_{01}) = r_i(x_i,d_i^*) + F1(st,t_i(x_i,d_i^*),x_{01}) = \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F1(st,t_i(x_i,d_i),x_{01}));$$

E. (* stage j *)

st = (j-1) mod 2;

dt = j mod 2;

For x_j=1 to k_j do

$$F(dt,x_j) = r_j(x_j,x_{01},d_j) + F1(st,t_j(x_j,x_{01},d_j),x_{01}));$$

$$= \max_{\substack{1 \le d_j \le p_j \\ 1 \le x_{01} \le k_{01}}}(r_j(x_j,x_{01},d_j) + F1(st,t_j(x_j,x_{01},d_j),x_{01}));$$

$$t_j(x_j,x_{01}^*,d_j^*) = t_j(x_j,x_{01}^*,d_j) + k_d;$$

F. (* stages j+1 through n *)

For i = j+1 through n do

$$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i)) = \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

$$t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d;$$

G. (* Find optimal return $F_n(x_n^*)$ *)

$$F_n = F(dt,x_n^*) = \max_{1 \le x_n \le k_n} F(dt,x_n);$$

H. (* optimal decisions from n through 1 *)

1. dt = s mod 2;

   For x_s=1 to k_s do

   $$F(dt,x_s) = r_s(x_s,d_s^*) + FS(t_s(x_s,d_s^*)) + FM(t_{s1}(x_s,d_s^*),x_{01}^*)$$

   $$= \max_{1 \le d_s \le p_s}(r_s(x_s,d_s) + FS(t_s(x_s,d_s) + FM(t_{s1}(x_s,d_s),x_{01}^*));$$

   $t_s(x_s,d_s^*) = t_s(x_s,d_s^*) + k_d$;

2. For i=s+1 to j-1 do

   st = (i-1) mod 2;

   dt = i mod 2;

   For x_i=1 to k_i do

   $$F(dt,x_i) = r_i(x_i,d_i^*) + F(st,t_i(x_i,d_i^*))$$

   $$= \max_{1 \le d_i \le p_i}(r_i(x_i,d_i) + F(st,t_i(x_i,d_i)));$$

   $t_i(x_i,d_i^*) = t_i(x_i,d_i^*) + k_d$;

3. For i=n down to j+1 do

   Search x_i-row of t_i for $t_i(x_i^*,d_i^*) \ge k_d$, $1 \le d_i \le p_i$;

   $x_{i-1}^* = t_i(x_i^*,d_i^*)$ mod k_d;

4. Search x_j^*-row of t_j for $t_j(x_j^*,x_{01}^*,d_j^*) \ge k_d$, $1 \le d_j \le p_j$;

   $x_{j-1}^* = t_j(x_j^*,x_{01}^*,d_j^*)$ mod k_d;

5. For i = j-1 down to 1 do

   Search x_i^*-row of t_i for $t_i(x_i^*,d_i^*) \ge k_d$, $1 \le d_i \le p_i$;

   $x_{i-1}^* = t_i(x_i^*,d_i^*)$ mod k_d;

   If i = s+1 then XS = x_s^* and DS = d_s^*;

I. (* Optimal returns, stages m_1 through 11 *)

1. For x_{11}=1 to k_{11} do

   $$F(1,x_{11}) = r_{11}(x_{11},d_{11}^*) = \max_{1 \le d_{11} \le p_{11}}(r_{11}(x_{11},d_{11}):t_{11}(x_{11},d_{11}) = x_{01}^*);$$

   $t_{11}(x_{11},d_{11}^*) = t_{11}(x_{11},d_{11}^*) + k_d$;

2. For i=2 to m do

   st = (i-1) mod 2;

   dt = i mod 2;

   For x_{i1}=1 to k_{i1} do

   $$F(dt,x_{i1}) = r_{i1}(x_{i1},d_{i1}^*) + F(st,t_{i1}(x_{i1},d_{i1}^*)) = \max_{1 \le d_{i1} \le p_{i1}}(r_{i1}(x_{i1},d_{i1}) + F(st,t_{i1}(x_{i1},d_{i1})));$$

   $$t_{i1}(x_{i1},d_{i1}^*) = t_{i1}(x_{i1},d_{i1}^*) + k_d;$$

3. $x_{m1}^* = t_{s1}(XS,DS)$

4. For i=m down to 1 do

   Search x_{i1}^*-row of t_{m1} for $t_{m1}(x_{i1}^*,d_{i1}^*) \ge k_d$

   For $1 \le d_{i1} \le p_{i1}$;

   $x_{i-1,1} = t_{i1}(x_{i1}^*,d_{i1}^*) \bmod k_d$;

5.3 *Storage Complexity of FBLCA*

All notations and techniques of section 2 are used.

Theorem 7: The storage complexity of FBLCA is given by $2k^2 + 2k$ or $O(k^2)$.

Proof: F1(0:1,1:k,1:k) are used to process the branch and after storing the optimal return at m-th stage in FM(1:k,1:k) both these tables can be released. Then F(0:1,1:k) are used to process main branch, and FS(1:k) for storage at stage for later use. So at any one time no more than $2k^2 + k$ storage spaces are used.

5.4 Computational Complexity of FBLCA

We use the notations and techniques of section 2.

Theorem 8: The computational complexity of FBLCA is given by

$$\sum_{i=1}^{m}(4k_{i1}k_{01}p_{i1} + 4k_{i1}p_{i1} - k_{i1}k_{01} + k_{i1} + p_{i1} + 12) + \sum_{i=j}^{s}(4k_i k_{01}p_i - k_i k_{01} + 5)$$

$$+ \sum_{i=1}^{n}(4k_i p_i + k_i + p_i + 7) + k_s p_s k_{01} - 4k_j p_j + k_s p_s + k_j k_{01} + k_{m1}k_{01} + k + k_{s-1}$$

$$+ 3k_n + 1/2(3_j^2 - 3_j - 34)$$

Proof: Number of computations at each step is listed below.

Step	Number of Computations
A.1	$4k_{11}k_{01}p_{11} - k_{11}k_{01}$
A.2	$\sum_{i=2}^{m}(4k_{i1}k_{01}p_{i1} - k_{i1}k_{01} + 5)$
A.3	$k_{m1}k_{01}$
B.1	k
B.2	$\sum_{i=1}^{s-1}(4k_i p_i + k_i + 5)$
B.3	k_{s-1}
C	$(5k_i k_{01}p_i - k_i k_{01} + 5)$
D	$\sum_{i=s+1}^{j-1}(4k_i k_{01}p_i - k_i k_{01} + 5)$
E	$4k_j k_{01}p_j + k_j + 5$
F	$\sum_{i=j+1}^{n}(4k_i p_i + k_i + 5)$
G	$3k_n - 1$
H.1	$5k_s p_s + k_s + 2$
H.2	$\sum_{i=s+1}^{j-1}(4k_i p_i + k_i + 5)$

H.3	$\mathop{}\limits_{i=j+1}^{n}(p_i + 2)$
H.4	$p_j + 2$
H.5	$\mathop{}\limits_{i=1}^{j-1}(p_i + 5)$
I.1	$4k_{11}p_{11} + k_{11}$
I.2	$\mathop{}\limits_{i=2}^{m}(4k_{i1}p_{i1} + k_{i1} + 5)$
I.3,4	$\mathop{}\limits_{i=1}^{m}(p_{i1} + 2) + 2$

Addition of these results gives the formula of the theorem.

<u>Corollary 11</u>: Suppose the discretization level of all state and decision variables are the same i.e, $k_{i1} = p_{i1} = k_i = p_i = k$ for admissable values of i. Then the formula for complexity becomes

$$(2m^2 + 2j^2 - 25^2 + 2m + 2j + 25 + 1)k^3 + 1/2(3m^2 + 4n^2 + s^2 - j^2 + 3m + 4n - j - s - 2)k^2 + (m^2 + n^2 + m + n + 5)k + 1/2(12m^2 + 7n^2 + 8j^2 - 5s + 2j - 34)$$

<u>Corollary 12</u>: The computational complexity of FBLCA can be described as $O(k^3)$, $O(m^2)$, $O(n^2)$.

<u>Corollary 13</u>: Suppose that all three subbranches in feedback loop system are of approximately equal lengths, i.e, $s = j/2 = n/3$. Then computational complexity formula reduces to

$$(8m^2 + 8m + 1)k^2 + (18m^2 + 6m - 1)k^2 + (10m^2 + 4m + 7)k + (51m^2 + 21m - 18) .$$

Thus this complexity has representations $O(k^3)$ and $O(m^2)$.

Corollary 14: The computational complexity depends on junction points of the loop. More specifically it is $O(j^2 - s^2)$ in terms of junction parameters.

6. Comparison and Analysis of Complexities

6.1 Analysis of Storage Complexities

We list the results from various network systems below in terms of parameter k where k is the level of discretization of state and decision variables.

System	Complexity	Depends on Junctions
Diverging Branch	$O(k)$	NO
Converging Branch	$O(k^2)$	NO
Feedforward Loop	$O(k^2)$	NO
Feedback Loop	$O(k^2)$	NO

6.2 Analysis of Computational Complexities

Results of various systems are listed below in terms of discretization level k, lengths of branch and main branch, and dependence on junctions points.

Parameter	k	m	n	Junction Points j and s
Converging Branch	$O(k^2)$	$O(m^2)$	$O(n^2)$	———
Diverging Branch	$O(k^3)$	$O(m^2)$	$O(n^2)$	———
Feedforward Loop	$O(k^3)$	$O(m^2)$	$O(n^2)$	$O(j^2 - s^2)$
Feedback Loop	$O(k^3)$	$O(m^2)$	$O(n^2)$	$O(j^2 - s^2)$

LME

8